Computers & Operations Research 78 (2017) 80–93

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Breakout local search for the multi-objective gate allocation problem

Una Benlic [a,c,*], Edmund K. Burke [a], John R. Woodward [b]

[a] School of Electronic Engineering and Computer Science, Queen Mary University of London, London, United Kingdom
[b] CHORDS Research Group, University of Stirling, Stirling FK9 4LA, United Kingdom
[c] University of Electronic Science and Technology of China, North Jianshe Road, Sichuan 610054, China

ARTICLE INFO

Article history:
Received 14 February 2015
Received in revised form
20 August 2016
Accepted 22 August 2016
Available online 30 August 2016

Keywords:
Gate allocation
Airports
Breakout local search
Greedy constructive heuristic
Adaptive diversification
Metaheuristics

ABSTRACT

The problem of assigning gates to arriving and departing flights is one of the most important problems in airport operations. We take into account the real multi-criteria nature of the problem by optimizing a total of nine gate allocation objectives that are oriented both on convenience for airport/airline services and passenger comfort. As far as we are aware, this is the largest number of objectives jointly optimized in the GAP literature. Given the complexity of the considered problem, we propose a heuristic approach based on the Breakout Local Search (BLS) framework. BLS is a recent variant of the Iterated Local Search (ILS) with a particular focus on the perturbation strategy. Based on some relevant information on search history, it tries to introduce an appropriate degree of diversification by determining adaptively the number and type of moves for the next perturbation phase. Moreover, we use a new memory-based greedy constructive heuristic to generate a starting point for BLS. Benchmark instances used for our experiments and comparisons are based on information provided by Manchester Airport.

1. Introduction

Due to the increase in the volume of air traffic, techniques for effective management of airport resources have gained an ever-increasing interest for both stakeholders in airports and academics. There are several major classes of decisions which need to be made by airline and airport management: ground operations scheduling, aircraft sequencing, crew assignment, deicing scheduling, etc. Nevertheless, one of the most important and complex airport related problems is the allocation of aircraft to gates. The term *gate* is often used in the literature to refer to terminal stands or off-pier stands on the apron, while we use the term *aircraft* to designate the arrival of an aircraft until the following departure of the same aircraft.

The gate allocation problem (GAP) consists of finding an allocation of aircraft (or rather of aircraft serving flights) to a limited number of gates (i.e., stands), provided that a set of hard and soft constraints is satisfied. Hard constraints must be strictly satisfied. If at least one of these constraints is violated, the solution is infeasible. The inherent hard constraints to GAP are that one gate can only accommodate a single aircraft at a time, and that two aircraft with overlapping times must not be allocated to the same gate. The assigning of each operation to a compatible gate has recently proven to be NP-complete [16]. Some additional hard constraints may also need to be considered, such as the space restriction related to the size of available gates and aircraft, and the shadowing restriction [12,13]. Recently, some effort has been made in [27] to integrate gate allocation with the ground movement problem by introducing a new constraint which limits the number of aircraft that are expected to block each other while manoeuvring in the area close to the gates.

Soft constraints are used to define the objective function to be optimized, and can be classified into passenger-oriented and airport-oriented. Different airports prioritize different objectives. Some airports give higher priority to the maximization of passenger comfort while others prioritize airport-oriented objectives. Moreover, conflicting objectives are very common. The vast majority of research on GAP focuses on minimization of the total passenger walking distance [1,11,17,26] to improve customer satisfaction. Optimization of this GAP objective is NP-hard because of its relationship to the classic quadratic assignment problem [28]. Other objectives considered in the literature are the increase of schedule robustness [4,8,27,29], the minimization of aircraft towing activities [12,13,21,29], the maximization of total aircraft-gate preferences [12,27,29], effective usage of gate space [27], minimization of unassigned aircraft [11,22,29], the minimization of taxi delay caused by push-back and taxi blocking [20,19], etc.

The earliest literature mostly considered the gate allocation problem as a single objective problem, and focused mainly on different passenger comfort objectives. Recent research considers a more realistic representation of GAP by taking into account a

* Corresponding author at: School of Electronic Engineering and Computer Science, Queen Mary University of London, London, United Kingdom.
E-mail addresses: u.benlic@qmul.ac.uk (U. Benlic), e.burke@qmul.ac.uk (E.K. Burke), jrw@cs.stir.ac.uk (J.R. Woodward).

http://dx.doi.org/10.1016/j.cor.2016.08.010

trade-off between three or more conflicting objectives. For instance, Dorndorf et al. [12] omitted the walking distance objective and put focus on the following three objectives: maximization of the total aircraft to gate preferences, minimization of the number of towed aircraft, and minimization of the deviation form the initial allocation. In [13], the authors consider four objectives: the maximization of the total assigned preference score, minimization of the number of unassigned aircraft, minimization of the number of tows, and also maximization of the schedule robustness with respect to aircraft delays. In a recent work, Kumar and Bierlaire [29] present mathematical formulations of different types of business and operational constraints, some of which have not been considered before in the GAP literature. Moreover, the authors jointly optimize a total of six objectives encountered at the Chicago O'Hare Airport: minimization of the cost for allocating aircraft to unfavorable gates, minimization of the towing activities, minimization of ungated aircraft, maximization of connection revenue, zone gate maximization, and maximization of the schedule robustness.

Given this wide range of GAP formulations and models, a variety of optimization techniques, both exact and heuristic, have been used. The choice of method depends on the GAP optimization model, i.e., the objectives and constraints considered. Earlier exact methods [1,4,31] are generally based on the branch-and-bound framework and its variations. Several recent exact approaches are based on MIP formulations using standard solvers [16,25]. In particular, Guépet et al. show how a natural MIP formulation can be strengthened with clique constraints, leading to significantly reduced MIP sizes and solution times below a minute for instances with up to 700 operations per day. In their model, they take into account two objectives, the maximization of the number of passengers/aircraft at contact stands and minimization of the number of towing movements, while respecting a set of operational and commercial requirements. More recently, Yu et al. [32] considered a GAP with three objectives: robustness, tow cost and transfer distance. To make the problem solvable, they transformed the quadratic model into an equivalent MIP model, and proposed a Variable Reduce Neighborhood Search (VRNS) algorithm which is a variant of a MIP-based heuristic for NP-hard models. Given the complexity of GAP formulations, many researchers use heuristic/metaheuristic methods including tabu search [7,8,30], simulated annealing [7,11], genetic algorithms [5,7,18,15], and hybrids [7,11,22]. This is often the case for GAP formulations entailing a variant of the passenger walking distance objective [11,17,26,30] which is NP-hard given its quadratic expression. For a detailed survey of GAP state-of-art, the interested reader is referred to [6,12].

In this paper, we take into account the real multi-criteria nature of the GAP problem by optimizing a linear combination of nine objectives that are oriented both on convenience for airport/airline services and passenger comfort. As far as we are aware, this is the largest number of objectives jointly optimized in the GAP literature. These can be grouped into four main categories: (i) idle times between conflicting aircraft, (ii) flight/aircraft to gate preference, (iii) tows and (iv) passenger walking distances. Given the complexity of our GAP formulation, we propose a Breakout Local Search (BLS) algorithm for this problem. BLS is a recent variation of Iterated Local Search (ILS) [23] which puts particular focus on the importance of the diversification phase. For each perturbation phase, it tries to introduce the most suitable degree of diversification into the search by determining dynamically the number of perturbation moves (i.e., the jump magnitude) and by adaptively choosing between two types of perturbations of different intensities. This is achieved through the use of information from the search history. In addition to GAP, BLS has been shown to provide competitive performance for several well-studied combinatorial problems, e.g., the maximum cut [2]. We further propose a new memory-based greedy constructive heuristic to generate a starting point for BLS. Benchmark instances used for our experiments and comparisons are based on information provided by Manchester Airport. Experimental comparisons show the benefit of BLS for GAP.

The paper is organized as follows. The problem description and formulation are presented in Section 2. Section 3 details the proposed BLS approach and the new greedy constructive procedure for GAP. Experimental results and comparisons on data from Manchester Airport are provided in Section 4. Finally, conclusions are presented in Section 5.

2. Gate allocation considering ground movement

The gate allocation problem considered in this paper consists of assigning a certain number of aircraft activities to a limited number of gates provided that a set of hard constraints is satisfied, while minimizing a linear combination of nine GA objectives that can be grouped into four categories. Before formally presenting GAP, we first provide the problem description and define the terms used throughout the paper.

2.1. Problem description

As mentioned previously, we use the term *aircraft* to refer to the arrival of an aircraft until the following departure of the same aircraft. Beside arrival and departure times, each aircraft is associated with additional specifications: the origin and destination of a aircraft, the type of aircraft, the aircraft registration number, the number of passengers and transferring passengers, the airline operating the aircraft, etc. We adopt the term *gate* to refer to both terminal stands and remote stands. A *terminal stand* is an area adjacent to a terminal building where an aircraft can easily be loaded and unloaded, while a *remote stand* is an area which is not attached to a terminal where an aircraft can park. The remote stands are used when all the terminal stands are occupied or if requested by airline operator.

It is uncommon for every aircraft to be able to use every gate. A gate might be too small for a given aircraft type or it may not have appropriate facilities to accommodate the aircraft. In addition, an aircraft cannot be allocated to a gate with inappropriate security facilities, e.g., domestic flights do not require the same level of security measures as international flights. In some cases, two adjacent gates cannot be used simultaneously, i.e., usage of one gate blocks usage of the other gate. More precisely, a large gate can sometimes be used as either two smaller gates (for two smaller aircraft) or as one gate for a large aircraft. In the literature, this type of constraint is referred to as a *shadowing restriction*. Fig. 1 shows an example where a large gate $G4$ is modeled as three gates: $G4F$ (full size gate), $G4L$ (the left side of gate $G4$) and $G4R$ (the right side of $G4$). The two gates $G4L$ and $G4R$ could be used simultaneously by two small aircraft $F2$ and $F4$, but neither $G4L$ and $G4R$ could be used if there is a large aircraft $F3$ parked at $G4F$.

One crucial issue of gate allocation is that some aircraft stay at the airport for an extended period of time, e.g., they arrive early in the morning and leave late in the evening. An aircraft is considered as a *long-stay aircraft* if the estimated time of its stay at the airport is longer than a fixed time limit. In the case of many such long-stay aircraft, the number of available terminal stands quickly decreases. To avoid this situation, gate planners have the possibility of splitting the stay of a long-stay aircraft between two or more gates, and towing the aircraft to a remote stand. In our formulation of GAP, we assume that an aircraft may need to be towed to another stand (generally a remote one), which incurs a certain

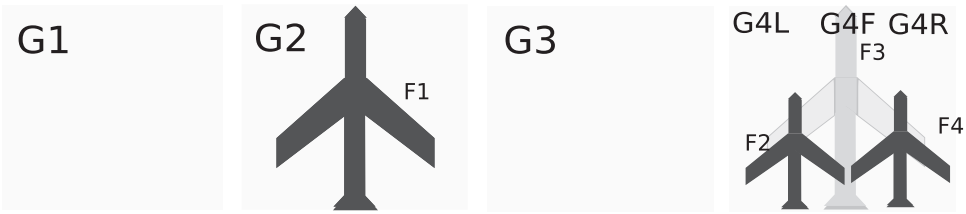


Fig. 1. An example where a large gate *G4* is used as two smaller gates *G4L* and *G4R* that can accommodate two small aircraft.

penalty. We also assume that a long-stay aircraft may not need to change gates if the airport is not particularly busy at the given time. As explained in the next section, the stay of each long-stay aircraft is thus split into three time intervals or parts that we simply refer to as aircraft activities. We distinguish between four types of aircraft activities:

Arrival: After the aircraft lands, it stays at the gate for not less than a fixed time limit (depending on the time required to unload), after which it might be towed to some other gate if required.

Tow: During this part, the aircraft resides on a gate other than that of its arrival and departure for at least a fixed period of time.

Departure: The aircraft is taken to its departure gate, for at least a fixed period of time before its departure.

Arrival/departure: If an aircraft needs to stay at airport for less than a given time limit, we consider it as a single activity.

In the best case, all parts (i.e., aircraft activities) are allocated together to one gate. In the worst case, all aircraft activities are assigned to different gates.

Another major airport operation problem is the routing of departure aircraft from gates to runways and arrival aircraft from runways to gates. There are usually several taxiways that can be used to get from one point to another, and the chosen taxiway is typically the shortest one without conflicts. An example of gates and taxiways around the gates in provided in Fig. 3.

Motivated by the work presented in [27], we consider the expected traffic at taxiways around the gates in order to reduce aircraft delays due to push-back and taxiway blockages. *Push-back blocking* arises when a departing aircraft is ready to leave a gate but is unable to since another aircraft is blocking its push-back trajectory. This form of blockage is only common in the areas close to gates. An example of push-back blocking is illustrated in Fig. 2a, where aircraft $F1$ is pushing back into the path of aircraft $F2$. One of these two aircraft needs to wait until the other has completed the manoeuver. *Taxi blocking* arises either when two aircraft are going in opposite directions along a taxiway as shown in Fig. 2b, or when two taxiways cross and two aircraft happen to be at the crossing point at the same time. *Taxi blocking* may not occur only around the gates but also further away from the gates on the way to/from the runway. It is particularly common where multiple taxiways converge. Our approach can only handle push-back and taxiway blockages in the areas around the gates. To identify possible blockages, we divide gates into groups as shown in Fig. 3. A *gate group* consists of a contiguous subset of gates that are reached using similar routes. These groups are not disjoint since one gate can we part of at most two gate groups. One of the objectives considered in our work is to maximize the idle time between activities that arrive (depart) to (from) gates that are from the same gate group.

2.2. Gate allocation problem formulation

We present a formulation of GAP considering ground movement around the gates. To introduce the hard and soft constraints, we state:

- the set F of aircraft;
- the set G of gates;
- the set SH of neighboring gate pairs that cannot be used simultaneously due to shadow restriction;
- the set GR of gate subsets belonging to the same gate group;
- the set A of all aircraft activities obtained by splitting all long-stay aircraft $\forall f \in F,\ (dep(f) - arr(f)) > t$ (where t=240 min) into three time intervals (i.e., activities). In our experiments, the duration d_1 of the first (arrival) activity of a long-stay aircraft is set to 60 min. After this period, the aircraft can be towed to another stand (generally a remote stand). The duration d_3 of the third (departure) activity depends on the aircraft size (aircraft size ranges from 1 to 12 according to the wingspan, 12 being the maximal aircraft size): $d_3 = 60$ for smaller aircraft (of size $\leq$5); $d_3 = 75$ for medium aircraft (of size six); and $d_3 = 105$ for large aircraft (of size $\leq$12). The duration d_2 of the second activity of a long-stay aircraft f is thus $d_2 = dep(f) - arr(f) - d_1 - d_3$. An aircraft that stays at airport for 240 min or less is considered as a single activity (arrival/departure). An example of splitting aircraft into aircraft activities is provided in Fig. 4.

We take $x_{a,g} = 1$ to denote that activity $a \in A$ is assigned to gate $g \in G$, and otherwise $x_{a,g} = 0$. Constants and variables are presented in Table 1, where the constants are predefined by the problem instance or regulated by airport management.

We consider the following four hard constraints H_1 to H_4 that are inherent to GAP:

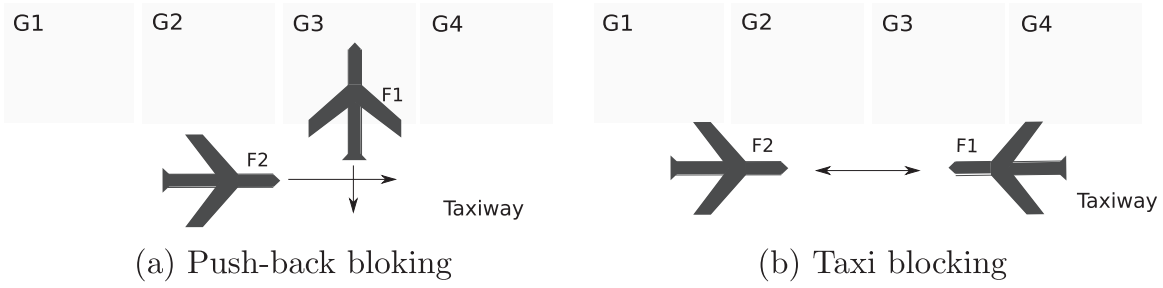


Fig. 2. Conflicts near gates.

Fig. 3. Taxiways and groups of gates at Manchester Airport.

H_1. *Restricted gates*: An activity $a \in A$ can only be allocated to a subset of possible gates:

$$x_{a,g} = 0, \quad \forall\, a \notin A(g), \quad \forall\, g \in G$$

In this work, we only consider gate restrictions due to gate size as well as exceptions when a gate of larger size cannot accommodate an aircraft of a smaller type. We do not consider gate restrictions due to security requirements since some airports do not require this constraint, i.e., a gate can accommodate an aircraft regardless of its origin and destination. However, gate restrictions due to security requirements can easily be integrated in our formulation.

H_2. *Single gate per activity:* Each activity $a \in A$ must be assigned to exactly one gate:

$$\sum_{g=1}^{|G|+1} x_{a,g} = 1, \quad \forall\, a \in 1, \ldots, |A|$$

A "dummy gate" (gate $|G| + 1$) is included in this constraint to represent the tarmac where aircraft arrive at when no gates are available.

H_3. *Activity overlap restriction:* No two activities can be allocated to the same gate at the same time:

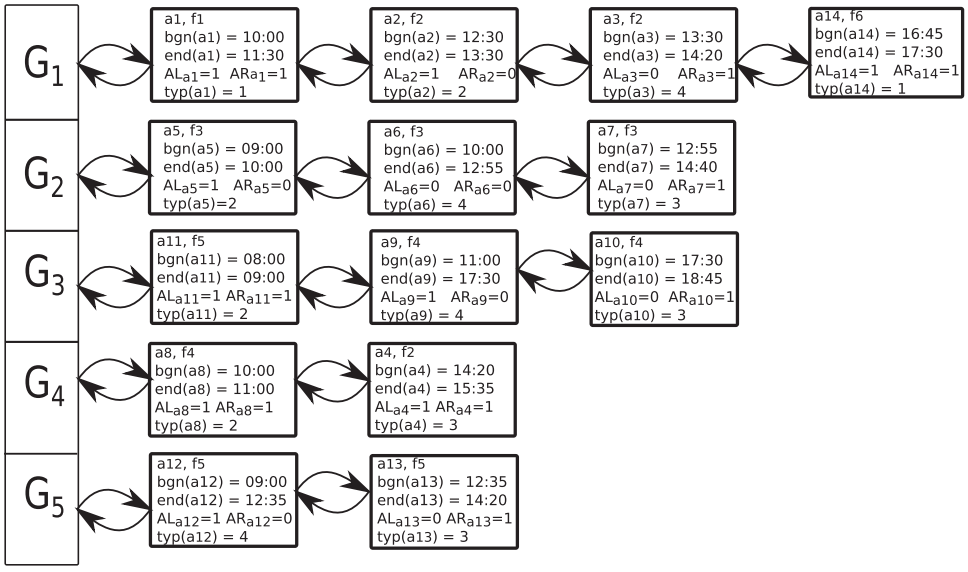
$$x_{a_1,g}x_{a_2,g}(end(a_2) - bgn(a_1))(end(a_1) - bgn(a_2)) \leq 1, \forall\, g \in G, \quad \forall\, a_1, a_2 \in A(g), \quad a_1 \neq a_2$$

H_4. *Shadow restriction*: Any two adjacent gates $(g_1, g_2) \in SH$ from the shadowing set cannot be used simultaneously:

$$x_{a_1,g_1}x_{a_2,g_2}(end(a_2) - bgn(a_1))(end(a_1) - bgn(a_2)) \leq 1, \forall\, (g_1, g_2) \in SH, \quad \forall\, a_1 \in A(g_1), \quad \forall\, a_2 \in A(g_2), \quad a_1 \neq a_2$$

Soft constraints are used to define the objective function to be optimized, and are associated to an activity-gate pair. We group the problem soft constraints into the following four classes:

(1) *Idle times between conflicting aircraft*: From this category, we



f_1: 10:00 - 11:30 --> a_1: 10:00 - 11:30
f_2: 11:30 - 15:35 --> a_2: 12:30 - 13:30; a_3: 13:30 - 14:20; a_4: 14:20 - 15:35
f_3: 09:00 - 14:40 --> a_5: 09:00 - 10:00; a_6: 10:00 - 12:55; a_7: 12:55-14:40
f_4: 10:00 - 18:45 --> a_8: 10:00 - 11:00; a_9: 11:00 - 17:30; a_{10}: 17:30 - 18:45
f_5: 08:00 - 14:20 --> a_{11}: 08:00 - 09:00; a_{12}: 09:00 - 12:35; a_{13} 12:35 - 14:20
f_6: 16:45 - 17:30 --> a_{14}: 16:45 - 17:30

Fig. 4. An example of an allocation of 14 activities (6 aircraft) to 5 gates.

Table 1
Constants and variables used in the formulation.

Constant	Description
$arr(f)$	The expected arrival time of aircraft $f \in F$
$dep(f)$	The expected departure time of aircraft $f \in F$
$bgn(a)$	The begin time of activity $a \in A$
$end(a)$	The end time of activity $a \in A$
$A(g)$	The subset of activities that can be allocated to gate $g \in G$
$A(f)$	The set of three activities of a long-stay aircraft $f \in F$, $A(f) = \{a_1, a_2, a_3\}$ (such that $bgn(a_1) = arr(f)$, $end(a_3) = dep(f)$) or a single activity of a short-stay aircraft
$SH(g)$	The subset of gates that cannot be used at the same time as gate $g \in G$ due to shadowing restriction
$GR(g)$	The set of gates that belong to the same gate group as $g \in G$
$fg(a)$	Number of feasible gates for activity $a \in A$, i.e., $fg(a) = \|\{g: a \in A(g)\}\|$
$isRS(g)$	Takes the value 1 if $g \in G$ is a remote stand, and 0 otherwise
$typ(a)$	Takes the value 1 if $a \in A(f)$, $bgn(a) = arr(f)$ and $end(a) = dep(f)$; 2 if $a \in A(f)$, $bgn(a) = arr(f)$ and $end(a) \neq dep(f)$; 3 if $a \in A(f)$, $bgn(a) \neq arr(f)$ and $end(a) = dep(f)$; and 4 otherwise
$pax\_arr(a)$	Total number of passengers associated to activity $a \in A(f)$ of $f \in F$, such that $bgn(a) = arr(f)$
$pax\_dep(a)$	Total number of passengers associated to activity $a \in A(f)$ of $f \in F$, such that $end(a) = dep(f)$
$ntx(a)$	Total number of transfer passengers associated to activity $a \in A(f)$ of $f \in F$, such that $bgn(a) = arr(f)$
$tx(a_1, a_2)$	Number of passengers from aircraft activity $a_1 \in A(f_1)$, $bgn(a_1) = arr(f_1)$ transferring to aircraft activity $a_2 \in A(f_2)$, $end(a_2) = dep(f_2)$.
$gtg(g_1, g_2)$	Estimated time required for passengers to transit from gate g_1 to gate g_2
$oper(a)$	Airline operator of activity $a \in A$
$pr_{oper(a),g}$	Preference rank of airline operating activity $a \in A$ for gate $g \in G$ (based upon statistical data analysis)
$npr_{oper(a)}$	Number of distinct preference ranks of airline operating activity $a \in A$
Variable	**Description**
AL_a	Takes the value 1 if two consecutive activities a and a' of the same aircraft are not allocated to the same gate (i.e., if $\exists f \in F$, $a \in A(f)$ and $a' \in A(f)$; $bgn(a) = end(a')$; $\exists g \in G$, $x_{a,g} = 1$ and $x_{a',g} = 0$), or if $a \in A(f)$ is an arrival activity, i.e., $arr(f) = bgn(a)$. Otherwise, $AL_a = 0$
AR_a	Takes the value 1 if two consecutive activities a and a' of the same aircraft are not allocated to the same gate (i.e., if $\exists f \in F$, $a \in A(f)$ and $a' \in A(f)$; $end(a) = bgn(a')$; $\exists g \in G$, $x_{a,g} = 1$ and $x_{a',g} = 0$), or if $a \in A(f)$ is a departure activity, i.e., $dep(f) = end(a)$. Otherwise, $AR_a = 0$
$p(a, g)$	The immediate previous activity of activity $a \in A$ at $g \in G$, i.e., $p(a, g) = a' \in A$: $\exists g \in G$, $x_{a,g} = 1$ and $x_{a',g} = 1$ and $\max_{end(a'):end(a') \leq bgn(a)}$. If a is the first activity allocated to g, $p(a, g) = -1$
$n(a, g)$	The immediate next activity of activity $a \in A$ at gate $g \in G$, i.e., $n(a, g) = a' \in A$: $\exists g \in G$, $x_{a,g} = 1$ and $x_{a',g} = 1$ and $\min_{bgn(a'):bgn(a') \geq end(a)}$. If a is the last activity allocated to g, $n(a, g) = -1$

take into account three soft constraints, with the objective to maximize idle times between aircraft that may be in conflict – (i) at the same gate (constraint S_1), (ii) at gates for which a shadowing constraint applies (constraint S_2), and (iii) at gates within the same gate group (constraint S_3).

Function $c_{a,g,1}$ assigns a penalty for violating a constraint of type S_1. Given an activity $a \in A(f)$ of aircraft f allocated to a gate g, $c_{a,g,1}$ assigns a positive cost to a considering the idle time $L_1(a, g)$ between a and its immediate previous activity $p(a, g)$ at g and/or the idle time $R_1(a, g)$ between a and its immediate next activity $n(a, g)$ at g, provided that $p(a, g)$ and/or $n(a, g)$ are not part of the same aircraft f, i.e., $p(a, g), n(a, g) \notin A(f)$:

$$c_{a,g,1} = \left\{ \begin{array}{cc} z(L_1(a,g)) + z(R_1(a,g)), & \text{if } (AL_a = 1 \wedge AR_a = 1) \\ z(L_1(a,g)), & \text{if } (AL_a = 1 \wedge AR_a = 0) \\ z(R_1(a,g)), & \text{if } (AL_a = 0 \wedge AR_a = 1) \\ 0 & \text{otherwise} \end{array} \right\} L_1(a,g)$$
$$= bgn(a) - end(p(a,g))R_1(a,g) = bgn(n(a,g)) - end(a)$$

To value the idle time, we adopt a cost function $z(time)$ proposed in [9] which is based on the *arctangent*, where *time* is the idle time. This function reflects the appreciation when any improvement is made for small idle times, whereas for large idle times any extra increase is of minor importance. In other words, it penalizes very small idle times with very high cost while mildly penalizing rather large idle times. The objective of $c_{a,g,1}$ is thus to minimize the value returned by $z(time)$:

$$z(time) = \arctan(0.21(5 - time)) + \frac{\pi}{2} \quad (1)$$

In case if $time = inf$ (inf is a very large number), $z(time)$ returns a value close to 0.

Similarly, the penalty function $c_{a,g,2}$ allocates a positive cost for violation of a constraint S_2. More precisely, it associates a cost to each activity a at gate g with respect to the minimal idle times $L_2(a, g)$ and/or $R_2(a, g)$ between a and activities allocated to $g' \in SH(g)$, provided that $SH(g) \neq \emptyset$:

$$c_{a,g,2} = \left\{ \begin{array}{cc} 0, & \text{if } (AL_a = 0 \wedge AR_a = 0) \vee (SH(g) = \emptyset) \\ z(L_2(a,g)) + z(R_2(a,g)), & \text{if } (AL_a = 1 \wedge AR_a = 1) \\ z(L_2(a,g)), & \text{if } (AL_a = 1 \wedge AR_a = 0) \\ z(R_2(a,g)), & \text{if } (AL_a = 0 \wedge AR_a = 1) \end{array} \right\} L_2(a,g)$$
$$= min\{inf, min\{(bgn(a) - end(a')): \forall g' \in SH(g), a' \in A, x_{a',g'} = 1, end(a') < bgn(a)\}\}R_2(a,g)$$
$$= min\{inf, min\{(bgn(a'') - end(a)): \forall g' \in SH(g), a'' \in A, x_{a'',g} = 1, bgn(a'') > end(a)\}\}$$

The aim is once again to minimize the cost function $z(time)$ (see Eq. (1)).

Similar objectives to S_1 and S_2, with the purpose to increase solution robustness, were considered in [4,13,10,29,22,27].

Finally, the purpose of S_3 is to maximize the amount of idle time between aircraft activities which can be in conflict (push-back or taxi blocking) if assigned to the same gate group. The penalty function $c_{a,g,3}$ assigns a penalty to each activity a at gate g by considering the minimal idle times $L_2(a, g)$ and/or $R_2(a, g)$ between a and activities allocated to gates $g' \in GR(g)$:

$$c_{a,g,3} = \left\{ \begin{array}{cc} z(L_3(a,g)) + z(R_3(a,g)), & \text{if } (AL_a = 1 \wedge AR_a = 1) \\ z(L_3(a,g)), & \text{if } (AL_a = 1 \wedge AR_a = 0) \\ z(R_3(a,g)), & \text{if } (AL_a = 0 \wedge AR_a = 1) \\ 0, & \text{otherwise} \end{array} \right\} L_3(a,g)$$

$$= \min\{inf, \{\min(|bgn(a) - bgn(a')| \times AL_{a'}, |bgn(a) - end(a')| \times AR_{a'}) : \forall\, g' \in GR(g), a' \in A, x_{a',g'} = 1\}\} R_3(a,g)$$

$$= \min\{inf, \{\min(|end(a) - bgn(a')| \times AL_{a'}, |end(a) - end(a')| \times AR_{a'}) : \forall\, g' \in GR(g), a' \in A, x_{a',g} = 1\}\}$$

To valorize the idle time returned by $L_3(a,g)$ and/or $R_3(a,g)$, we once again use the cost function $z(time)$ from Eq. (1).

In [27], the authors introduce for the first time a hard constraint which limits the number of aircraft which are expected to block each other if assigned to gates from the same group. Furthermore, the minimization of taxi delay, caused by push-back and taxi blocking, has been considered in [20,19].

(2) *Flight/aircraft to gate preference*: This category considers four types of soft constraints whose objectives are (i) to maximize the usage of gate space (constraint S_4), (ii) to maximize airline preferences for a particular gate (constraint S_5), (iii) to minimize the number of tows to terminal gates (constraint S_6), and (iv) to minimize the number of passengers arriving or departing from remote gates (constraint S_7).

The aim of S_4 is to avoid allocating unnecessarily large gates since these have more flexibility for use by aircraft which have to be moved on the day of operations. Ideally, the aircraft size of activity a allocated to gate g should be equal to the maximal size *maxSize* that g can accommodate. Otherwise, $c_{a,g,4}$ associates a penalty depending on the difference between *maxSize* and the size of aircraft serving activity a. The larger this difference is, the higher is the penalty:

$$c_{a,g,4} = (\text{maxSize}(g) - size(a)) \cdot \text{maxSize}(g)$$

Effective usage of gate space has previously been considered in [27].

The preferences of airlines for particular gates are determined based upon a statistical data analysis which examines how often particular gates have been used by particular airlines. The preference rank $pr_{oper(a),g}$ indicates that gate g is the $pr^{th}_{oper(a),g}$ most frequently used gate by $oper(a)$, and can take the value from 0 to $npr_{oper(a)} - 1$. If two gates are used by $oper(a)$ for the same number of times, we assign them the same rank. The penalty function $c_{a,g,5}$ associated to S_5 can be expressed as

$$c_{a,g,5} = \frac{pr_{oper(a),g}}{npr_{oper(a)} - 1}$$

Maximization of airline-gate preferences was also addressed in [12,13,29,27].

Given that a common practice at airports is to use remote stands for towed aircraft, we further take into account constraints S_6 with the aim to reduce the number of towed aircraft at terminal gates. However, note that towing an aircraft to a terminal stand may in some cases reduce the possibility of traffic congestions on the way to a remote stand. For this reason, we consider towing to a terminal stand as a soft constraint and associate a positive penalty to a tow activity a if the corresponding aircraft is parked at a terminal stand g:

$$c_{a,g,6} = \left\{ \begin{array}{cc} 1, & \text{if}(typ(a) = 4 \wedge AL_a = 1 \wedge AR_a = 1 \wedge isRS(g) = 0) \\ 0, & \text{otherwise} \end{array} \right\}$$

Finally, the purpose of S_7 is to reduce the number of passengers that need to take a bus or walk to and from remote stands. For each arrival and departure activity a allocated to a gate g, the penalty function $c_{a,g,8}$ assigns a cost which depends on the total number of passengers that need to load or unload at a remote stand, where *maxPax* is the maximal total number of passengers at any flight. Moreover, the penalty associated to an arrival aircraft activity depends on the number of transfer passengers that need to take a bus or walk to the terminal building, where *maxTxp* is the maximal number of transfer passengers at any flight:

$$c_{a,g,7} = \left\{ \begin{array}{cc} \frac{pax\_arr(a)}{maxPax} + \frac{ntx(a)}{maxTxp} + \frac{pax\_dep(a)}{maxPax}, & \text{if } (isRS(g) = 1 \wedge typ(a) = 1) \\ \frac{pax\_arr(a)}{maxPax} + \frac{ntx(a)}{maxTxp}, & \text{if } (isRS(g) = 1 \wedge typ(a) = 2) \\ \frac{pax\_dep(a)}{maxPax}, & \text{if } (isRS(g) = 1 \wedge typ(a) = 3) \\ 0, & \text{otherwise} \end{array} \right\}$$

(3) *Tows*: The objective associated to soft constraint S_8 is the minimization of the number of tow (park) activities and the number of gate changes. The function $c_{a,g,8}$ pushes activities of the same long-stay aircraft $f \in F$, $|A(f)| > 1$ to be allocated to the same gate g by assigning a penalty to activity $a \in A(f)$ if activities $a' \in A(f)$, $end(a') = bgn(a)$ and/or $a'' \in A(f)$, $end(a) = bgn(a'')$ are not allocated to g:

$$c_{a,g,8} = \left\{ \begin{array}{cc} 2, & \text{if } (a, a', a'' \in A(f)) \wedge (bgn(a) = end(a')) \\ & \quad \wedge (end(a) = bgn(a'')) \wedge (x_{a',g} = 0) \\ & \quad \wedge (x_{a'',g} = 0) \\ 1, & \text{if } (a, a', a'' \in A(f)) \wedge (bgn(a) = end(a')) \\ & \quad \wedge (end(a) = bgn(a'')) \\ & \quad \wedge ((x_{a',g} = 1) \oplus (x_{a'',g} = 1)) \\ 0, & \text{otherwise} \end{array} \right\}$$

Minimization of the number of towing activities is also considered in [12,13,16,21,29].

(4) *Passenger walking distances*: We aim to reduce the possibility of passenger missing a connecting flight by minimizing the estimated time required for transfer passengers to go from one gate to another (the corresponding soft constraint is labeled as S_9). The penalty function c_{a_1,a_2,g_1,g_2} assigns a positive cost to the arrival and the departure activities a_1 and a_2 allocated to gates g_1 and g_2 respectively in terms of the time $gtg(g_1, g_2)$ needed to go from gate g_1 to g_2, the number of passengers transferring from a_1 to a_2, and the time difference between the arrival and connecting flight activities. *MaxGtGT* and *maxTxp* refer respectively to the maximal time needed to go from one gate to another, and the maximal number of transfer passengers on any flight. The function z from Eq. (1) ensures a very high penalty for short times between connections, while only mildly penalizing long times between connections. Transfer passengers that have more than 120 min between a connecting flight are not taken into account:

$$c_{a_1,g_1,a_2,g_2,9} = \left\{ \begin{array}{cc} \frac{gtg(g_1,g_2) \cdot tx(a_1,a_2)}{maxTxp \cdot maxGtGT} \cdot z(end(a_2) - bgn(a_1)), & \text{if } (end(a_2) > bgn(a_1)) \wedge (typ(a_1) \neq 4 \wedge typ(a_2) \neq 4) \wedge (end(a_2) - bgn(a_1)) \leq 120 \\ \frac{gtg(g_2,g_1) \cdot tx(a_2,a_1)}{maxTxp \cdot maxGtGT} \cdot z(end(a_1) - bgn(a_2)), & \text{if } (end(a_1) > bgn(a_2)) \wedge (typ(a_1) \neq 4 \wedge typ(a_2) \neq 4) \wedge (end(a_1) - bgn(a_2)) \leq 120 \\ 0 & \text{otherwise} \end{array} \right\}$$

Different variants of the passenger walking distance objective

have been extensively considered in the GAP literature [1,11,17,26]. Because of its relationship to the classic quadratic assignment problem [28], optimization of this GAP objective is NP-hard and hence the considered formulation of GAP is NP-hard.

Weighted sum of several objectives is a commonly used procedure in the literature for evaluation of the total cost of a GAP solution. Given the formulation we have just described, we thus compute the total soft constraint cost for a given candidate feasible solution $\mathcal{S}$ with the cost function $c(\mathcal{S})$ defined in Eq. (2),

$$c(\mathcal{S}) = \sum_{a=1}^{|A|}\sum_{g=1}^{|G|}\sum_{i=1}^{8} x_{a,g}c_{a,g,i}w_i + \sum_{a_1=1}^{|A|}\sum_{a_2=1}^{|A|}\sum_{g_1=1}^{|G|}\sum_{g_2=1}^{|G|} x_{a_1,g_1}x_{a_2,g_2}c_{a_1,g_1,a_2,g_2,9}w_9 \quad (2)$$

where w_i is the weight associated to the soft constraint S_i, regulated by the airport operations management. Note that different weights may be assigned to different soft constraints so as to produce solutions that are more convenient for their particular needs. A weight could be 0 if the corresponding soft constraint is not considered. The problem objective is then to find a feasible solution $\mathcal{S}^*$ such that $c(\mathcal{S}^*) \leq c(\mathcal{S})$ for all $\mathcal{S}$ in the feasible search space.

3. Breakout local search (BLS)

Given the complexity of our GAP formulation and given the nonlinearity of the objective function defined in Eq. (2), we propose a breakout local search (BLS) and a greedy constructive heuristic for GAP, detailed in the following sections.

3.1. General framework of BLS

BLS is a recent stochastic local search method [2,3] which follows the general scheme of iterated local search (ILS) [23]. Its basic idea is to use a descent-based local search procedure to explore in depth the current search space region, and to diversify the search once a local optimum is attained. More precisely, BLS triggers an adaptive and multi-typed perturbation mechanism to introduce a suitable degree of diversification required to escape from the current local optimum. As explained in [2,3], the degree of diversification introduced with BLS depends on the number L of perturbation moves (also called jump magnitude) and on the type T of perturbation moves (e.g., random perturbation or directed perturbation). Algorithm 1 provides a general framework for BLS, which is explained in the previous works [2,3].

Algorithm 1. BLS general framework.

```
1: S'← GenerateInitialSolution
2: L ← L_0 /*Initialize the number L of perturbation moves */
3: while stopping condition not reached do
4:   S ← DescentBasedSearch(S')
5:   L ← DetermineJumpMagnitude(L, S, history)
6:   T ← DeterminePerturbationType(S, history)
7:   S' ← Perturb(L, T, S, history)
8: end while
```

We next detail the solution representation and the four main component procedures of our BLS algorithm for GAP.

3.2. Solution representation

For an effective search with BLS, a gate allocation S is represented as an array of $|G|$ lists (one list for each $g \in G$), where each list S_g consists of a number of aircraft activities allocated to g. Each activity $a \in S_g$ is associated with some information including the begin time $bgn(a)$, the end time $end(a)$, and the aircraft number corresponding to the given activity. Activities in each S_g are ordered by their begin and end times, i.e., $\forall\, a \in S_g$, $bgn(a) \geq end(p(a))$ and $end(a) \leq bgn(n(a))$, where $p(a)$ and $n(a)$ are the immediate previous and immediate next activities of a respectively. An example of a solution with 5 gates and 14 activities (corresponding to 6 aircraft) is shown in Fig. 4.

3.3. Initial solution – greedy constructive heuristic

We use a memory-based greedy constructive heuristic (MGCH) to generate a feasible starting point for the search, i.e., an initial allocation S^0 of all the aircraft activities from A to a set of gates G, such that all the hard constraints H_1 to H_4 are satisfied. The MGCH algorithm is presented in Algorithm 2.

Algorithm 2. Memory-based greedy constructive heuristic (MGCH).

```
ENSURE: A feasible initial solution (allocation) S^0
1: P ← ∅
2: S^0 ← ∅
3: iter ← 0
4: counter[|A|] ← set all values to 0
5: for i:=1 to |A| do
6:   g ← selectBestFeasibleGate(a_i)
7:   if (g ≠ −1) /*a_i can feasibly be allocated to g*/ then
8:     S^0_g ← a_i /*Allocate activity a_i to g*/
9:     iter ← iter + 1
10:  else
11:    P ← backtrack(a_i)/* A feasible gate allocation in the
       current partial solution does not exist, backtracking re-
       quired */
12:    while (P ≠ ∅) do
13:      a ← firstElementfromList(P)
14:      P ← P\{a}
15:      counter[a]:=counter[a] + 1
16:      g ← selectBestFeasibleGate(a)
17:      if (g ≠ −1) then
18:        S^0_g ← a_i
19:        iter ← iter + 1
20:      else if counter[a] < κ then
21:        P ← P + backtrack(a)/* Insert the deallocated ac-
           tivities to P*/
22:      else
23:        counter[|A|] ← set all values to 0
24:      end if
25:    end while
26:  end if
27: end for
```

Let $c(a, g)$ be the sum of the penalties S_1 to S_9 (see Section 2.2) for placing activity $a \in A$ to gate $g \in G$. For each $a \in A$, MGCH first calls the function $selectBestFeasibleGate(a)$ (line 6) to determine a $g \in G$ for $a \in A(g)$ such that all the hard constraints H_1 to H_4 are satisfied and the cost $c(a, g)$ with respect to the partial solution S^0

is minimized. If a feasible allocation for a exists in the current partial solution ($g \neq -1$), MGCH allocates a to g (lines 7 and 8). Otherwise, MGCH calls the *backtrack*(a) procedure (line 11) which deallocates a given number of activities from a gate g_b, $a \in A(g_b)$, in order to make place for the critical activity a. These deallocated activities are placed into the set P. Let R be the set of activities that need to be removed in the backtrack phase to insert a critical element a, the backtrack procedure determines a gate g_b for a with the following relation:

$$g_b \in \{g' : a \in A(g'), \max\left\{\sum_{a' \in R} c(a', g')/|R| - c(a, g')\right\}, tabu(a, g') = 0\} \quad (3)$$

where $tabu(a, g)$ is a function which returns 1 if the allocation of activity a to gate g is prohibited, and 0 otherwise. Move prohibition is determined in the following way. Each time an activity is allocated to gate g, it can be deallocated without restriction. However, it is forbidden to reallocate a to g for a given number of iterations *iter* (tabu tenure). More precisely,

$$tabu(a, g) = \begin{Bmatrix} 1, & \text{if } (\gamma(a, g) + \alpha_1 \cdot fg(a) + random(\alpha_2 \cdot fg(a))) \geq iter \\ 0, & \text{otherwise} \end{Bmatrix}, \quad (4)$$

where $\gamma(a, g)$ is the matrix that keeps track of the iteration number when activity a was last allocated to g, $fg(a)$ is the number of feasible gates for a, and α_1 and α_2 are two coefficients. Prohibition of certain allocations in the backtrack phase reduces the number of unsuccessful reallocations (i.e., the number of ungated activities). Note that the described backtrack procedure is based on ideas borrowed from the tabu search metaheuristic [14].

After allocation of a to g_b has been completed, MGCH tries to reallocate the activities from P (lines 12–25) as follows. Starting from the first activity a in P, MGCH first calls the function *selectBestFeasibleGate*(a) to find the best feasible allocation of a to some gate g. If such an allocation does not exist, MGCH then checks whether the number of trials to allocate a (*counter*[a]) has exceeded a certain threshold κ ($\kappa = 100$ in our experiments). If *counter*[a] > κ, activity a is left ungated and is not considered for allocation any more. Otherwise, MGCH calls the backtrack procedure to allocate a by removing a certain number of activities from S^0 which are added to the set P. These steps are repeated until $P = \emptyset$.

3.4. Neighborhood move and local search

To move from one solution to another in the search space, our BLS for GAP applies an insert move which consists in moving a single aircraft activity $a \in A(g)$ from its current gate g to another eligible gate g', $a \in A(g')$. However, this move may violate hard constraints H_3 and/or H_4 leading to an infeasible solution. In that case, the resulting solution is repaired by deallocating a number of activities in conflict with activity a, and reallocating them to other gates as performed during the greedy construction phase (lines 11–25 of Algorithm 2).

The purpose of the descent-based local search procedure of BLS is to intensify the search in the current search region. Each iteration of the descent-based local search consists in selecting a highest penalty activity-gate pair (a,g) in the current schedule S, and moving a to another gate g_n. To select the highest penalty activity-gate (a, g) pair in the current solution in $O(1)$, we keep each (a,g) pair ordered by their costs. This is achieved with an adaptation of bucket sort data structure [2] that is commonly used for graph partitioning problems. A new gate g_n for a is selected according to the following relation:

$$g_n \in \left\{ g' : a \in A(g'), \max\left\{\sum_{a' \in R} c(a', g')/|R| - c(a, g')\right\}\right\}, \quad (5)$$

where R is the set of activities, currently allocated to gate g', that have to be removed from the current solution and reallocated to other gates (as explained in the previous subsection) in order to allocate activity a.

The descent-based local search procedure stops as soon as a local optimum is encountered.

3.5. Adaptive diversification strategy

To determine the most suitable number of perturbation moves at a given stage of the search, the proposed BLS for GAP takes advantage of the information related to the occurrences of cycles. This information is based on a number of the most recently visited locally optimal solutions stored in a hash table structure (HT), as performed in [3]. Moreover, for each perturbation phase, BLS employs a probabilistic technique to select between directed and critical element-guided perturbation moves, the former being based on history information maintained in a recency-based tabu list [14]. These two types of perturbations introduce different degrees of diversification into the search. The probability of determining one perturbation type over another depends on the current search state, i.e., the current number of consecutive non-improving local optima visited with respect to a reference solution.

We next describe in detail the proposed diversification mechanism, along with the perturbation types used.

3.5.1. Determining jump magnitude

After a local optimum S is attained during the descent-based local search phase, BLS determines a suitable number of moves for the next perturbation phase. This procedure is given in Algorithm 3.

BLS calls the function *inHashTable* (line 1 of Algorithm 3) to check whether S is already stored in the hash table HT memory. If S is not in HT, *inHashTable* returns 0 and inserts S into HT. Otherwise, *inHashTable* returns 1. BLS increments the number of perturbation moves L if the search keeps consecutively returning to an already visited local optimum for a fixed number of times ν (lines 3–5 of Algorithm 3), and decreases L if a new local optimum is reached (lines 6–9 of Algorithm 3). Finally, we limit the number of perturbation moves to take values no smaller than L_{MIN} (lines 10–12 of Algorithm 3).

Algorithm 3. DetermineJumpMagnitude (L, S, HS, *counter*).

Require: Current jump magnitude L, local optimum S returned by *DescentBasedSearch*, hash table memory HS, and the number of consecutive encounters of an already visited solution *counter*
Ensure: Jump magnitude L for the next perturbation phase
1: **if** (*inHashTable*(HS, S) = 1) **then**
2: *counter* ← *counter* + 1
3: **if** (*counter* > ν) **then**
4: $L \leftarrow L + 1$ /* Increment the jump magnitude*/
5: **end if**
6: **else**
7: $L \leftarrow L - 1$ /* Decrement the jump magnitude */
8: *counter* ← 0
9: **end if**
/* Limit L to take values no smaller than L_{MIN} */
10: **if** $L < L_{MIN}$ **then**
11: $L \leftarrow L_{MIN}$
12: **end if**

Table 2
Description of benchmark instances.

Inst.	$\|F\|$	$\|A\|$	#s-s	#l-s	% ogpt	Inst.	$\|F\|$	$\|A\|$	#s-s	#l-s	% ogpt
D_1	295	519	183	112	75.78	D_{1-6}	1780	2908	1216	564	82.11
D_2	300	540	180	120	81.05	*PK*0.87	307	535	193	114	88.42
D_3	291	539	167	124	72.63	*PK*0.97	315	543	201	114	96.84
D_4	300	548	176	124	76.84	*PK*1.03	322	550	208	114	103.15
D_5	323	579	195	128	80.00	*PK*1.14	333	571	214	119	113.68
D_6	270	498	156	114	76.84						

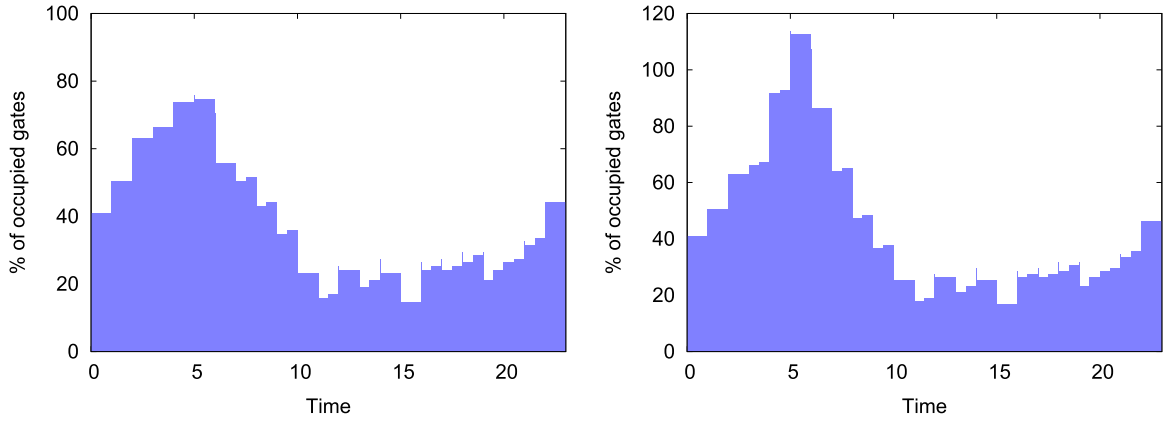


Fig. 5. Histogram showing the percentage of occupied gates over one day of operations at Manchester Airport.

13: return L

3.5.2. Adaptive combination of two perturbation types

To perturb the current local optimum S, BLS adaptively chooses between directed perturbation and critical element-guided perturbation.

The *directed perturbation* (or the tabu-based perturbation) is based on the tabu search principles [14]. It consists in selecting a highest penalty activity-gate pair (a,g) in the current allocation, and moving a to another gate $g_b \neq g$. The selection of g_b depends both on the quality of the move in order not to deteriorate too much the perturbed solution, and the history information which keeps track of the last iteration (time) when the given move was performed. More precisely, a g_b for a is determined with the relation defined in Eq. (3).

Let K be the set of the first λ highest penalty activity-gate pairs, where λ is a parameter. The *critical element-guided perturbation* (CEGP) [24] first consists in randomly selecting a pair $(a, g) \in K$ and deallocating a from g. The procedure then sorts all the feasible gates g', such that $a \in A(g')$, in a decreasing order according to the penalty function value:

$$penalty(a, g') = \frac{\sum_{a' \in R} c(a', g')}{|R|} - c(a, g'),$$

where R is once again the set of activities that have to be removed from g' in order to allocate a to g', and $c(a, g')$ is the cost associated to the assignment of a to g' expressed as the sum of the soft constraint penalties S_1–S_9. Finally, activity a is allocated to one of the possible gates in an adaptive and random way, according to the penalty score assigned to the gate – the higher the penalty value, the more chance the gate is chosen for allocation. For this purpose, the rth highly-scored gate is selected according to the following probability function:

$$Prob(r) = r^{-\phi} / \sum_{i=1}^{fg(a)} i^{-\phi},$$

where ϕ is a positive real number (empirically set to $\phi \in [1.5, 3.0]$) which determines the intensity of the selection procedure. The larger the value of ϕ is, the higher is the possibility that the high-score gates are selected. Note that the classic random perturbation is a special case of the critical element-guided perturbation when $\phi = 0$.

We set the parameters for these two types of perturbations so that the directed perturbation is weaker than the critical element-guided perturbation.

In order to insure the best balance as possible between an intensified and a diversified search, BLS uses a strategy given in Algorithm 4 that takes turns probabilistically between the directed and the critical element-guided perturbation. The probability P of applying a particular perturbation is determined with relation $P = e^{-\omega/\eta}$ with respect to the search state ω, i.e., the current number ω of consecutive non-improving local optima visited with respect to a reference solution S_{ref}. The idea is to apply the directed perturbation with a higher probability P whenever the search progresses towards local optima with objective values better than that of S_{ref} (counter ω is small). With the increase of ω, the probability P of using directed perturbation decreases while the probability of applying the critical element-guided moves increases for the purpose of a stronger diversification. ω is reset to zero each time the reference solution S_{ref} is updated, or when the number of consecutive non-improving local optima exceeds a given threshold η (lines 3 and 6 of Algorithm 4). Solution S_{ref} is updated with a new local optimum S if $c(S)\cdot\mu$ is smaller than $c(S_{best})$, where μ is a coefficient that can take a value in the range [0, 1] (lines 1–4 of Algorithm 4).

Algorithm 4. *DeterminePerturbationType* $(S, S_{best}, S_{ref}, \omega)$.

Require: Local optimum S, best solution found during the

Table 3
Settings of important parameters and weights associated to soft constraints $S_1 - S_9$.

Parameter	Section	Description	Value
α_1	3.3, 3.5.2	Coefficient for tabu tenure	0.8
α_2	3.3, 3.5.2	Coefficient for tabu tenure	2.5
ν	3.5.1	Coefficient for increase of the number of perturb. moves	60
L_{MIN}	3.5.1	Minimal number of perturbation moves	5
μ	3.5.2	Coefficient for update of S_{ref}	0.99
λ	3.5.2	Coefficient for critical element-guided perturbation	0.2|A|
ϕ	3.5.2	Coefficient for critical element-guided perturbation	2.2
η	3.5.2	Coefficient for probability P	5000
w_1	2.2	Idle times between conflicting aircraft (S_1)	12
w_2	2.2	Idle times between conflicting aircraft (S_2)	12
w_3	2.2	Idle times between conflicting aircraft (S_3)	8
w_4	2.2	Aircraft to gate preference (S_4)	0.3
w_5	2.2	Aircraft to gate preference (S_5)	15
w_6	2.2	Aircraft to gate preference (S_6)	15
w_7	2.2	Aircraft to gate preference (S_7)	30
w_8	2.2	Tows (S_8)	5
w_9	2.2	Passenger walking distances (S_9)	40

search S_{best}, reference local optimum S_{ref}, number of consecutive non-improving local optima visited ω with respect to S_{ref}
Ensure: Perturbation type T.
if $(c(S)\cdot\mu < c(S_{best}))$ **then**
  $S_{ref} \leftarrow S$ /* Update reference local optimum*/
  $\omega \leftarrow 0$
end if
if $(c(S_{ref}) > c(S))$ or $(\omega > \eta)$ **then**
  $\omega \leftarrow 0$
else
  $\omega \leftarrow \omega + 1$
end if
Determine prob. P of applying directed over critical element-guided perturb.: $P = e^{-\omega/\eta}$
if $(P > random\{0, 0.01, 0.02, \ldots, 1\})$ **then**
  $T \leftarrow Directedperturbation$
else
  $T \leftarrow Criticalelement - guidedperturbation$
end if
return T

4. Experimental results

4.1. Problem instances

The data used in these experiments is provided by Manchester Airport and contains:

- information on aircraft over six days – it includes the estimated arrival/departure times, the number of passengers, flight operators, aircraft types, etc;
- information about gate usage, i.e., which gates cannot be used simultaneously due to shadowing restriction, and a list of aircraft types that can be allocated to the given gate;
- information on aircraft sizes that divides aircraft into groups from 1 to 12 according to their wingspan;
- actual (final) allocation of gates.

In short, Manchester Airport has three terminals and a total of 95 stands/gates (62 terminal stands and 33 remote stands). The size of the stands ranges from 2 to 12 (size 12 being the largest gate size). Out of the 95 stands, 21 stands induce a shadowing restriction, i.e., they can be used as either two smaller gates (for two smaller aircraft) or as one gate for a large aircraft.

Information that is not provided by Manchester Airport include:

- information related to transfer passengers, such as the total number of transfer passengers for each arrival and details on connecting flights;
- the estimated time required to go from one gate to another;
- gate preferences of airline operators;
- the division of gates into gate groups according to the path required to reach the gate.

We generate the missing data either by statistical analysis of the provided information or randomly, based on our understanding of the problem.

For our experiments, we use a set of 11 benchmark instances. Instances D_1–D_6 correspond to one day of planned flights at Manchester Airport, with the number of aircraft per day ranging from 270 to 323. Instance D_{1to6} includes all the aircraft from day 1 to day 6. Since the difficulty of a gate allocation depends mostly on the number of aircraft that need to be allocated at peak time (i.e., the percentage of gates allocated at peak time), we use four additional instances *PK* which are obtained by modifying instance D_1 by adding a certain number of additional aircraft during peak time. For each benchmark instance, Table 2 provides the number of aircraft and activities, the number of short-stay aircraft (column *#s-s*), the number of long-stay aircraft (column *#l-s*), and the percentage of gates occupied at peak time (column % *ogpt*). Note that % *ogpt* may be greater than 100% since, in some cases, a large gate may be used as two small gates. Histograms showing the percentages of used gates for instances *D*1 and *PK*1.14 over one day of operations are provided in Fig. 5.

4.2. Experimental protocol

Our BLS algorithm is implemented in C++ and compiled with GNU g++ under GNU/Linux running on an Intel Xeon E5335 with 2 GHz and 2 GB of RAM. The setting of BLS parameters, as well as the weights $w_1 - w_2$ (see Eq. (2)) corresponding to soft constraints $S_1 - S_9$, are provided in Table 3. Since different airports prioritize different objectives, the weights may be varied in an attempt to obtain solutions that are more appropriate for the particular needs. For each benchmark instance, we perform 30 independent

Table 4
Averaged computational results (in terms of objective value, see Eq. (2)) over all the benchmark instances and executions, obtained with BLS and seven other heuristics.

Avg. performance	TS	ILS-DIR	ILS-CEG	MGCH	ILS-I	ILS-II	ILS-III	BLS
Avg. best	13,194.7	13,177.5	14,954.4	14,954.4	12,981.9	13,048.9	13,549.3	**12,766.1**
Avg. mean	13,405.4	13,448.7	14,954.5	14,954.5	13,256.2	13,265.6	13,769.9	**12,995.4**
Avg. worst	13,883.0	13,876.6	14,955.2	14,955.2	13,502.7	13,445.2	13,957.2	**13,228.2**
Avg. time	2798.4	2876.4	0.0	0.0	3238.3	3317.0	3477.4	3207.7

Table 5
Post hoc analysis of the normalized performances reported with BLS and the reference algorithms across all the benchmark instances and executions.

	TS	ILS-DIR	ILS-CEG	MGCH	ILS-I	ILS-II	ILS-III	BLS
TS	–	–	–	–	–	–	–	–
ILS-DIR	0.00293	–	–	–	–	–	–	–
ILS-CEG	0.00098	0.00098	–	–	–	–	–	–
MGCH	0.00098	0.00098	–	–	–	–	–	–
ILS-I	0.04199	0.01855	0.00098	0.00098	–	–	–	–
ILS-II	0.00684	0.00488	0.00384	0.00098	0.32030	–	–	–
ILS-III	0.70020	0.70020	0.00098	0.00098	0.00293	0.00098	–	–
BLS	0.00098	0.00098	0.00092	0.00092	0.00195	0.00488	0.00098	–

executions with the time limit set to 1.5 h per run.

We compare the performance of BLS (both in terms of quality and computing time) with seven other heuristic algorithms:

MGCH: the greedy constructive procedure (see Section 3.3) which generates an initial solution for BLS;
TS: a tabu search algorithm which corresponds to the directed perturbation used by BLS (see Section 3.5.2). The parameters α_1 and α_2 for the tabu list are set to 0.8 and 2.5 respectively;
ILS-DIR: an iterated local search which combines the descent-based local search (see Section 3.4) with the directed (tabu-based) perturbation (see Section 3.5.2). The number of perturbation moves for ILS-DIR is fixed to 45. The search performed with this approach is highly oriented toward intensification;
ILS-CEG: an iterated local search approach which combines the descent-based local search with the critical element-guided perturbation (see Section 3.5.2). The number of perturbation moves for ILS-CEG is fixed to 4. The perturbation phase of ILS-CEG introduces a high degree of diversification into the search that can almost be compared to a random restart;
ILS-I: a modification of BLS obtained by fixing the probability P of applying the directed over the critical element-guided perturbation to 0.75;
ILS-II: a modification of BLS obtained by fixing the number of perturbation moves $L=100$;
ILS-III: a modification of BLS obtained by fixing L to 100 and P to 0.75.

The five latter algorithms are used to evaluate the importance of the adaptive and multi-type perturbation mechanism employed by our BLS algorithm for GAP. We apply MGCH to generate an initial solution for each local search algorithm used in this comparison. The results for this comparison are obtained under the same computing conditions.

We do not provide comparisons with the actual gate allocations provided by Manchester Airport since these are obtained with rules different than those applied in this work. More precisely, our solutions are based on incomplete data from Manchester Airport. Moreover, we do not know the exact objectives as well as the priorities of objectives for Manchester Airport. Therefore, any comparison with the actual allocations would be inaccurate and inconclusive.

4.3. Computational results and comparisons

This section presents computational results and comparisons between the proposed BLS and the seven heuristic approaches mentioned in the previous section.

For each approach, Table 4 shows the average best, the average and the average worst objective value (according to Eq. (2) over all the benchmark instances after 30 executions. The best (minimal) values are highlighted in bold. We further show the average time in seconds per run for each approach. From these results, we can make the following observations. The proposed BLS generally provides the best performance on the used benchmark compared to the seven reference heuristics. BLS and all the other local search algorithms except ILS-CEG are able to significantly improve the initial solution obtained with MGCH. Only ILS-CEG, which introduces a higher degree of diversification into the search than ILS-DIR, is unable to improve the starting solution obtained with MGCH. We may thus conclude that the critical element-guided perturbation, used by our BLS for GAP, is unable to direct the search towards promising regions of the search space.

To assess whether there exists a significant difference in the observed performances between at least two algorithms on our set of benchmark instances, we first normalize the obtained objective values into a common range of values. More precisely, for each approach, we obtain a set Y of normalized objective values over all the problem instances and executions. We then employ the Friedman rank sum test on all the sample sets Y. If significant difference in performances exists, we carry out a post hoc analysis to determine which two algorithms differ in performance. The null hypothesis of the Friedman rank sum test states that all normalized objective value samples have equal medians, while the alternative hypothesis states that there exist at least two samples with different median values. For the post hoc analysis, we perform pairwise Wilcoxon-signed rank tests on the normalized sample sets.

Since the Friedman rank sum test reveals a statistically significant difference in performance between the considered algorithms (with $p=1.609e-11$), we continue with the post hoc analysis and show in Table 5 the obtained p-values from the Wilcoxon-signed rank tests. The tests confirm that BLS statistically outperforms the reference algorithms with a p-value ≤ 0.00195. It also confirm that the combination of the directed and the critical element-guided perturbations is highly important for the performance of our BLS. Indeed, both ILS-I and ILS-II ensure a statistically better performance than TS, ILS-DIR and ILS-CEG (p-value <0.05). Furthermore, the adaptive BLS strategy for determining the number and the type of perturbation moves is another key factor for success of the proposed approach. Indeed, there is no difference in performance between ILS-III, TS and ILS-DIR according to the post hoc test.

For each instance, Table 6 summarizes the best solutions obtained by BLS in terms of the GAP objectives, using the penalty weights given in Section 4.2. To justify the need for local optimization, we further provide the individual objective values for the initial solutions generated with MGCH.

With the current setting of penalty weights, both BLS and MGCH are able to allocate all aircraft activities (even in the case of *PK*1.14) with a 100% success rate. For the optimized solutions, the minimal idle times between activities at the same gate range from 10 min to 15 min, while the minimal idle times between activities

Table 6
Comparison between the best solutions obtained with BLS and MGCH, in terms of the individual GAP objectives.

Objective	D_1		D_2		D_3		D_4		D_5		D_6	
	BLS	MGCH	BLS	MGCH	BLS	MGCH	BLS	MGCH	BLS	MGCH	BLS	MGCH
Min. (avg.) idle time between activities at the same gate (in min.)	10 (212.7)	10 (212.3)	**15** (199.6)	10 (163.7)	**15** (193.6)	10(182.3)	**15** (192.0)	10 (175.3)	10 (175.1)	10 (144.9)	10 (177.6)	10 (161.2)
Min. (avg.) idle time between activities at shadowing gates (in min.)	20 (503.9)	**25** (607.8)	40 (419.7)	40 (422.8)	**45** (458.4)	15 (363.7)	**30** (789.1)	15 (517.6)	20 (342.9)	20 (368.2)	**20** (412.9)	10 (411.9)
Min. (avg.) idle time between activities at the same gate group (in min.)	**5**(93.4)	0 (91.6)	**5** (85.7)	0 (59.2)	**5** (81.4)	0 (76.7)	**5** (85.5)	0 (65.5)	**5** (69.6)	0 (56.8)	**5** (83.3)	0 (76.9)
Avg. gate space used in excess	1.4	1.4	1.31	1.31	**1.11**	1.17	**1.14**	1.15	1.22	1.22	**1.19**	1.36
Avg. airline preferences for particular gates	**0.49**	0.51	**0.51**	0.56	**0.52**	0.57	**0.49**	0.58	**0.54**	0.61	**0.54**	0.57
Total number of tows to terminal gates	0	0	0	0	0	0	0	0	0	0	0	0
% pax arriving at remote gate	2.5	**0.9**	**4.0**	4.5	3.5	**2.9**	**3.8**	7.0	**2.7**	4.9	**1.7**	5.3
% pax departing from remote gate	8.7	**6.3**	10.6	**8.4**	9.1	**7.0**	7.6	**6.9**	8.5	**7.6**	7.8	**7.7**
% transfer pax arriving at remote gate	1.5	**0.9**	**2.7**	4.8	2.9	**2.6**	**3.7**	7.0	**2.7**	4.5	**1.6**	4.6
Total number of gate changes	**62**	103	**61**	77	**69**	91	**74**	76	**79**	88	80	**76**
Avg. number of gate changes per long-stay aircraft	**0.55**	0.92	**0.51**	0.64	**0.56**	0.73	**0.60**	0.61	**0.62**	0.69	0.70	**0.67**
% transfer pax missing a connecting flight	3.3	**2.2**	3.0	**2.1**	**0.6**	1.6	2.9	**1.8**	2.9	2.9	2.6	**1.2**
Number of ungated activities	0	0	0	0	0	0	0	0	0	0	0	0

Objective	D_{1-6}		*P*0.87		*P*.97		*P*1.03		*P*1.14	
	BLS	MGCH	BLS	MGCH	BLS	MGCH	BLS	MGCH	BLS	MGCH
Min. (avg.) idle time between activities at the same gate (in min.)	10 (321.6)	10 (323.3)	10 (204.9)	10 (207.9)	**15** (209.8)	10 (212.6)	10 (208.8)	10 (211.1)	10 (203.9)	10 (208.5)
Min. (avg.) idle time between activities at shadowing gates (in min.)	**15** (1633.7)	10 (1453.3)	**25** (583.7)	10 (554.9)	**20** (546.6)	15 (534.2)	**25** (491.9)	15 (533.8)	**25** (489.3)	10 (567.2)
Min. (avg.) idle time between activities at the same gate group (in min.)	**5** (65.1)	0 (47.1)	**5** (94.6)	0 (89.0)	**5** (80.9)	0 (84.7)	**5** (99.5)	0 (83.0)	**5** (77.3)	0 (79.9)
Avg. gate space used in excess	1.33	**1.32**	**1.23**	1.39	**1.29**	1.37	**1.26**	1.37	**1.39**	1.43
Avg. airline preferences for particular gates	**0.55**	0.59	0.52	0.52	0.53	0.53	0.56	**0.53**	0.55	**0.53**
Total number of tows to terminal gates	0	0	0	0	0	0	0	0	0	0
% pax arriving at remote gate	**3.5**	5.0	2.9	**1.4**	**2.8**	3.9	**3.2**	4.7	**2.8**	6.3
% pax departing from remote gate	8.7	**7.3**	9.9	**6.7**	13.1	**8.9**	13.3	**9.5**	15.0	**10.8**
% transfer pax arriving at remote gate	**2.8**	4.5	1.6	**0.9**	**1.6**	4.1	**1.9**	4.6	**2.0**	5.5
Total number of gate changes	**452**	522	**69**	107	**78**	103	**82**	103	**90**	114
Avg. number of gate changes per long-stay aircraft	**0.80**	0.93	**0.61**	0.94	**0.68**	0.90	**0.72**	0.90	**0.76**	0.96
% transfer pax missing a connecting flight	1.6	**1.1**	**1.9**	2.1	**2.4**	3.4	2.3	2.3	2.6	**2.0**
Number of ungated activities	0	0	0	0	0	0	0	0	0	0

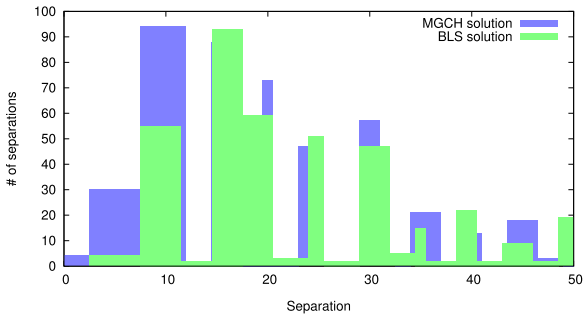


Fig. 6. A comparison of BLS and MGCH solutions in terms of idle times between aircraft activities allocated to the same gate group.

at shadowing gates range from 15 to 45 min. Moreover, these BLS solutions reduce the number of push-back and taxiway conflicts near the gates by introducing at least 5 min of separation between activities allocated at the same gate group. The initial solutions are slightly less robust, and would more likely increase the possibility of taxiway conflicts around the gates. Indeed, for each benchmark instance, the minimal idle time between activities allocated at the same gate group is 0 min in case of the MGCH solutions. Fig. 6 provides a comparison of BLS and MGCH solutions for instance D_1 in terms of idle times between aircraft activities allocated to the same gate group.

As for the effective gate usage, the average gate space used in excess is less than or equal to 1.4 for all the benchmark instances, in case of both the initial and the optimized solution. This can be explained by a relatively high penalty weight (w_4=0.3) associated to this objective. A preliminary tuning of penalty weights showed that the lower the value of w_4, the higher the number of ungated aircraft activities is, in case of full capacity during peak times. However, in case of regular days at Manchester Airport, a better compromise between the gate-space usage and the other objectives may be obtained by reducing w_4 (e.g., with w_4=0.05).

We further observe that the average satisfaction of airline preferences for specific gates ranges from 0.49 to 0.56 for optimized solutions (note that the value of this objective should ideally be close to 0). For MGCH, this value is often slightly higher, ranging from 0.51 to 0.61. The total number of passengers arriving and departing at/from a remote stand is always below 15%, for both BLS and MGCH solutions.

As for towing, a long-stay aircraft changes a gate from 0.51 to 0.7 times on average during a regular day (in our problem formulation, an aircraft can change gates at most 2 times) given a solution optimized with BLS. This implies that a significant number of long-stay aircraft stay at the same gate during their entire stay at the airport which reduces the total number of aircraft movements on the ground. On the other hand, the average number of gate changes for MGCH solutions is significantly higher, and ranges from 0.61 to 0.92 for instances D_1–D_6. Therefore, BLS solutions greatly reduce the number of aircraft movements which is an important objective for busy airports.

Finally, Table 6 shows that the percentage of passengers missing a connecting flight is often somewhat higher for the BLS than for the MGCH solutions, and ranges from 0.6% to 3.3% (1.1% to 2.9% for MGCH solutions). However, note that the provided percentages are based on generated data as mentioned in Section 4.1.

To summarize, BLS significantly improves the quality of the initial solution, particularly in terms of separation between aircraft at the same gate group and in terms of the number of aircraft movements on the ground.

5. Conclusion

In this work, we take into account the real multi-criteria nature of the gate allocation problem (GAP) and consider nine objectives that aim to optimize idle times between conflicting aircraft, aircraft to gate preference, aircraft towing and passenger walking distances. As far as we are aware, this is the largest number of objectives jointly optimized in the GAP literature. Given the non-linearity of the considered problem formulation, we propose a heuristic approach that follows the general framework of Breakout Local Search (BLS). BLS is able to obtain high quality gate allocations with reasonable computing efforts. Based on relevant information from its search history, it tries to establish a most suitable degree of diversification for each perturbation phase by determining dynamically the number of perturbation moves (i.e., the jump magnitude) and by adaptively choosing between the directed (tabu-based) and the critical element-guided perturbation. Moreover, we present a new memory-based greedy constructive heuristic (MGCH) to generate a starting point for the BLS search. To evaluate the performance of BLS, we perform statistical comparisons with a tabu search approach and five variants of ILS, using benchmark instances that are based on information provided by Manchester Airport. Experimental results demonstrate the usefulness of BLS for GAP, and accentuate the contribution of the multi-type perturbation adaptive perturbation mechanism to the performance of the proposed approach.

Acknowledgment

This work was partially supported by the research project EP/H004424/2. We are grateful to the anonymous referees for valuable suggestions and comments which helped us improve the paper.

References

[1] Babic O, Teodorovic D, Tosic V. Aircraft stand assignment to minimize walking. J Transp Eng 1984;110(1):55–66.
[2] Benlic U, Hao JK. Breakout local search for the max-cut problem. Eng Appl Artif Intell 2013;26(3):1162–73.
[3] Benlic U, Hao JK. Breakout local search for the vertex separator problem. Int J Conf Artif Intell (IJCAI) 2013, 461-467.
[4] Bolat A. Procedures for providing robust gate assignments for arriving aircrafts. Eur J Oper Res 2000;120(1):63–80.
[5] Bolat A. Models and a genetic algorithm for static aircraft-gate assignment problem. J Oper Res Soc 2001;52(10):1107–20.
[6] Bouras A, Ghaleb MA, Suryahatmaja US, Salem AM. The airport gate assignment problem: a survey. Sci World J 2014; 2014.
[7] Cheng CH, Ho SC, Kwan CL. The use of meta-heuristics for airport gate assignment. Expert Syst Appl 2012;39(16):12430–7.
[8] Şeker M, Noyan N. Stochastic optimization models for airport gate assignment. Transp Res Part E 2012;48:438–59.
[9] Diepen G, Van Den Akker JM, Hoogeveen JA, Smeltink JW. Using column generation for gate planning at Amsterdam Airport Schiphol. Technical Report, Institute of Information and Computing Sciences; 2007.
[10] Diepen G, van den Akker JM, Hoogeveen JA. Integrated gate and bus assignment at Amsterdam Airport Schiphol. Robust Online Large-Scale Optim 2009:338–53.
[11] Ding H, Lim A, Rodrigues B, Zhu Y. The over-constrained airport gate assignment problem. Comput Oper Res 2005;32(7):1867–80.
[12] Dorndorf U, Drexl A, Nikulin Y, Pesch E. Flight gate scheduling: state-of-the-art and recent developments. Int J Manag Sci 2007;35:326–34.
[13] Dorndorf U, Jaehn F, Pesch E. Modeling robust flight-gate scheduling as a clique partitioning problem. Transp Sci 2008;42(3):292–301.
[14] Glover F. Tabu search—part i. ORSA J Comput 1986;1(3):190–260.
[15] Gu Y, Chung CA. Genetic algorithm approach to aircraft gate reassignment problem. J Transp Eng 1999;125(5):384–9.
[16] Guépet J, Acuna-Agost R, Briant O, Gayon JP. Exact and heuristic approaches to the airport stand allocation problem. Eur J Oper Res 2015;246(2):597–608.
[17] Haghani A, Chen M. Optimizing gate assignment at airport terminals. Transp Res 1998;32A(6):437–54.
[18] Hu XB, Di Paolo EA. An efficient genetic algorithm with uniform crossover for

the multi-objective airport gate assignment problem. IEEE Congr Evolut Comput 2007:55–62.
[19] Kim SH. Airport control through intelligent gate assignment [Ph.D. thesis]. Georgia Tech; 2013.
[20] Kim SH, Feron E, Clarke JP, Marzuoli A, Delahaye D, Airport gate scheduling for passengers, aircraft, and operation. CoRR abs/1301.3535; 2013.
[21] Kumar P, Bierlaire M. Multi-objective airport gate assignment problem. In: Proceedings of the Swiss transport research conference; 2011.
[22] Lim A, Wang F. Robust airport gate assignment. Int Conf Tools Artif Intell 2005:74–81.
[23] Lourenco HR, Martin O, Stützle T. Iterated local search, handbook of meta-heuristics. Springer-Verlag, Berlin, Heidelberg; 2003.
[24] Lü Z, Hao JK. A critical element-guided perturbation strategy for iterated local search. Evolut Comput Comb Optim 2009:1–12.
[25] Maharjan B, Matis TI. Multi-commodity flow network model of the flight gate assignment problem. Comput Ind Eng 2012;63(4):1135–44.
[26] Mangoubi RS, Mathaisel DFX. Optimizing gate assignments at airport terminals. Transp Sci 1985;19(2):173–88.
[27] Neuman UM, Atkin JAD. Gate assignment considering ground movement. In: Lecture notes in computer science, vol. 8197; 2013. p. 184–198.
[28] Obata T. The quadratic assignment problem: evaluation of exact and heuristic algorithms. Technical Report, TS-7901, New York: Rensselaer Polytechnic Institute; 1979.
[29] Prem Kumar V, Bierlaire Michel. Multi-objective airport gate assignment problem in planning and operations. J Adv Transp 2014;48(7):902–26.
[30] Xu J, Bailey GT. The airport gate assignment problem: mathematical model and a tabu search algorithm. Hawaii Int Conf Syst Sci 2001:1–10.
[31] Yan S, Huo C. Optimization of multiple objective gate assignments. Transp Res 2001;35A(5):413–32.
[32] Yu C, Zhang D, Lau HYK. Mip-based heuristics for solving robust gate assignment problems. Comput Ind Eng 2016;93:171–91.